Tema

1

Introducción a la Informática Industrial

El uso de dispositivos electrónicos avanzados para automatización, control, y supervisión tanto en la Industria Civil como en la Militar, ha sido acompañado desde sus primeras etapas por sistemas programables de diferente complejidad. En este tema veremos algunos de los conceptos necesarios relativos a este tipo de desarrollos.

- Alcances y objetivos.
- Introducción a los Sistemas de Tiempo Real (STR).
- Características de los STR empotrados.

Alcances y objetivos

EN general las aplicaciones industriales realizan tareas de Automatización, Control y Supervisión. Dado el grado de complejidad que se exige actualmente, la solución basada exclusivamente en los clásicos sistemas mecánicos, hidráulicos, o electrónicos simples, no es suficiente. Casi la única solución factible es que sean desarrolladas usando dispositivos electrónicos programables. Este tipo de aplicaciones no pueden ser tratadas de la misma forma que otros desarrollos orientados más bien a la gestión de documentos como por ejemplo los programas ofimáticos. Las aplicaciones industriales requieren una respuesta correcta en el momento oportuno. Los sistemas que deben actuar teniendo en cuenta restricciones temporales más o menos estrictas, se llaman *Sistemas de Tiempo Real*.

Por *automatización* entendemos cualquier labor de actuación sobre una máquina que permita realizar labores sin la intervención humana directa. Para llevar a cabo una automatización efectiva, el ser humano ha tenido que perfeccionar su conocimiento de las leyes físicas, y matemáticas, lo que le ha permitido evolucionar desde las primeras máquinas simples como palanca, polea, y cuña, hacia máquinas complejas como relojes, bombas de agua, o trenes, con comportamientos hasta cierto punto, conocidos y predecibles. Cualquier sistema automático requiere disponer de alguna fuente de energía que permita realizar el trabajo sin que suponga esfuerzo a operario alguno, al menos en los instantes en los que se realiza dicho trabajo. En las numerosas ocasiones en que se requiera autonomía, la energía será acumulada química o mecánicamente mediante pilas, baterías, combustibles, muelles, resortes, etc.

El ser humano tiene una percepción simbólica y superficial de los agentes que le rodean asignándoles forma de sistemas, ya que disponen de unas entradas, y generan unas salidas. Además

observa que dichos sistemas mantienen un estado interno que puede evolucionar con el tiempo. El estado no es más que una instantánea de los atributos que describen las características internas del sistema. Casi cualquier ente físico puede ser considerado como un sistema atendiendo a determinados aspectos. A continuación se citan algunos ejemplos de sistemas simples y complejos, con las características de entrada/salida/estado relacionadas con su utilidad; estas se han elegido de forma deliberadamente caprichosa:

Tabla 1 Ejemplos de sistemas

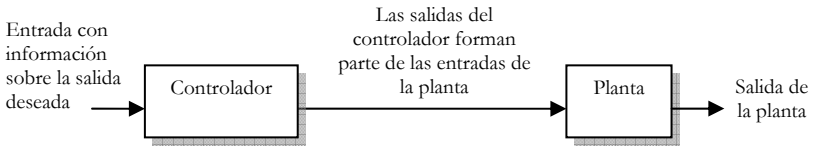
Sistema	Entrada	Salida	Estado
Piedra	Fuerza.	Posición.	Cantidad de movimiento.
Reloj despertador	Establecimiento en algún momento de la hora actual, y de la hora a la que debe despertar.	Una imagen que representa la hora actual, y un aviso acústico suficiente como para despertar.	La hora actual.
Caballo	Alimento, cobijo, descanso eventual.	Fuerza y resistencia para cambiar su posición y así remolcar de un carro, o ser montado por un jinete.	Velocidad, cansancio, hambre.
Automóvil	Combustible, comburente, acelerador, freno.	Fuerza, cambio de posición	Velocidad, desgaste interno

Normalmente no podemos tener una descripción completa de los sistemas debido a la complejidad intrínseca de los objetos reales, por lo que solemos abstraerlos y considerar tan solo las entradas, salidas, y atributos internos más relevantes para un propósito determinado. Igualmente la interpretación de estos datos como demuestra la tabla previa, puede ser arbitraria e incluso en ocasiones poco objetiva. Es importante observar que la mayoría de los sistemas están formados a su vez a partir de sistemas más simples, por lo que podrían disgregarse hasta llegar a niveles sólo describibles en términos de física cuántica. La definición de sistema es en cierto sentido recursiva, ya que cualquier sistema puede ser considerado parte integral de otro sistema mayor.

Los sistemas susceptibles de ser estudiados de forma científica han de tener entradas, salidas, y estados evaluables numéricamente. Tal es el caso de los sistemas en los que intervienen variables físicas, siendo también evaluables las variables aleatorias y otro tipo de estimaciones estadísticas[1], y aquellos sistemas que cuentan con variables discretas. La única restricción es que puedan tener una representación numérica, que de alguna forma pueda ser almacenada, clasificada y procesada.

Suele ocurrir que nos es de interés la capacidad de un sistema de generar una determinada salida, pero nos gustaría que lo hiciera con el objetivo de conseguir un fin determinado. Para ello y teniendo un conocimiento previo del comportamiento del mismo sería suficiente con aplicar una determinada entrada a sabiendas que esto implicaría la obtención de la salida requerida como una reacción natural. Al hecho de excitar la entrada de un sistema para conseguir una salida deseada lo conocemos con el nombre de *control*. La parte encargada de controlar es de por sí otro sistema. Con el objetivo de diferenciarlos, llamaremos *planta* al sistema controlado, y llamamos *controlador* al sistema encargado de generar las entradas de la planta.

[1] Téngase en cuenta que incluso lo métodos actuales de medición de magnitudes físicas, por muy exactos y/o precisos que sean, ofrecen resultados estadísticos que reflejan entre otras cosas la incertidumbre en la medida.



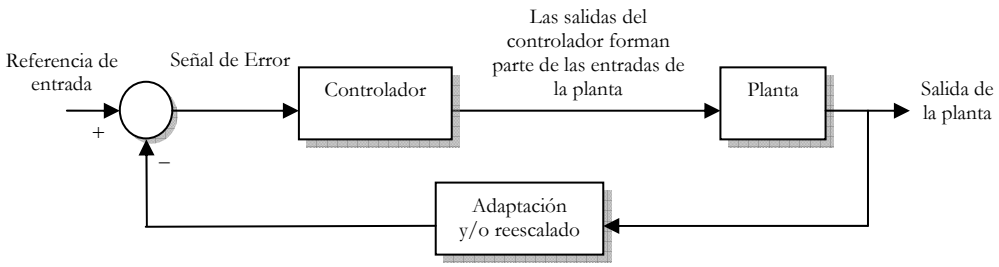
El esquema de control mostrado en la figura es el más empleado normalmente para la realización tareas relativamente simples. Se caracteriza por que el controlador no verifica que la salida de la planta toma el valor deseado, sino que se confía en que no surjan otros agentes externos (o internos) que pudieran desviar el funcionamiento que se había considerado. A este tipo de control se le denomina en *lazo abierto*, en contraposición al control en lazo cerrado que se comentará después. A continuación se citan algunos ejemplos de tales sistemas:

- **Tostadora de pan**.- El objetivo es calentar el pan que se le introduzca hasta que la superficie quede crujiente y tostada. La realidad es que la máquina no verifica que el pan quede como queremos, ni tampoco se asegura de que le hayamos metido pan. Si no hemos ajustado el controlador de acuerdo al tipo de pan es posible que quede poco hecho o incluso carbonizado.
- **Lavadora de ropa**.- Se pretende eliminar manchas y olores de la ropa. La máquina no verifica que esto sea así, ni siquiera tiene en cuenta el tipo de material que está lavando.
- **Motor paso a paso**.- Es un elemento habitualmente empleado en el control en lazo abierto ya que su controlador podría determinar tanto la posición del rotor, como las vueltas que ha dado desde que lo empezó a controlar. Sin embargo cualquier bloqueo del rotor que pudiera ser provocado por el encallamiento de algún engranaje, no sería tenido en cuenta por el controlador.
- **Elementos informativos**.- Normalmente los elementos como puedan ser LEDs, displays, LCDs, monitores, zumbadores, timbres, etc. encargados de informar sobre algún aspecto del estado interno de un sistema, son en sí mismos plantas controladas en lazo abierto. El controlador no sabrá por tanto si ha dejado de funcionar una parte de un display, o si se ha "fundido" un LED, con las consecuencias que esto podría acarrear.

Es una característica habitual en este tipo de sistemas que el controlador sea programado de alguna forma o al menos que la secuencia generada de excitación de la planta avance con arreglo a una temporización. Normalmente el controlador necesita algún tipo de ajuste inicial o periódico para que el control pueda realizarse empezando desde una situación conocida, siendo esta una característica necesaria. En caso contrario, el comportamiento podría degenerar.

La principal razón del uso de sistemas de control en lazo abierto es el abaratamiento, del controlador debido a su sencillez. No obstante sólo es válido cuando el comportamiento de la planta está completamente descrito, como en el caso de plantas simples, o cuando la verificación de los resultados sea tremendamente complicada. Un control de este tipo no deja de introducir cierta incertidumbre, al presuponer unos resultados que no son verificados, por lo que sólo se emplea con éxito para funciones no críticas, o consideradas no peligrosas. Por ejemplo, es muy difícil determinar por medios automáticos el grado de limpieza de la ropa que se metió en la lavadora, aunque desde el punto de vista de una persona sea una tarea trivial.

En el caso de querer asegurar que la salida está dentro de unos márgenes, es necesario comparar los datos que ofrece la planta con la referencia que pretendemos alcanzar. Tal circunstancia crea un bucle de control, por lo que decimos que se trata de un control en *lazo cerrado*.



Los datos de referencia son una representación de la salida que se querría conseguir, y a menudo están expresados en un formato adaptado a la entrada del controlador, habitualmente una tensión, o un número digital. Es por ello que sea necesaria una adaptación de la magnitud controlada de la que sólo necesitamos extraer la información que la caracteriza. Como este tipo de controladores necesitan captar información de la salida de la planta, es necesario que hagan uso de diferentes tipos de sensores que no aparecían en el control en lazo abierto. Obsérvese que la entrada al controlador es en este caso una señal de error, cuya información está relacionada con lo que faltaría para que la salida de la planta alcanzara el valor deseado. Tanto el hardware como la algorítmica de este tipo de control pueden llegar a ser muy complejos, lo que dificulta su desarrollo, y lo encarece.

A menudo los lazos de control son cerrados por un controlador humano, ya que llegado a cierto nivel sería muy difícil evaluar automáticamente y de forma eficaz los datos de entrada a los sistemas, debido sobre todo a la tremenda complejidad analítica que pueden llegar a alcanzar. Por ejemplo, hasta el día de hoy sigue siendo necesario que los automóviles sean conducidos por personas, ya que si no fuera así, el dispositivo necesario para cerrar el lazo de control debería ser capaz de hacer un tratamiento demasiado enrevesado de la información visual y acústica del entorno. Además se presentarían situaciones imprevistas, que tendrían que ser resueltas con sentido común, cosa que todavía no hemos sido capaces de programar.

El conjunto de sistemas controlador-planta forman por definición otro sistema. Tengamos en cuenta que un sistema así construido podría por ejemplo, ser considerado parte del controlador de otra planta *(!!!)*.

Existe también una amplia parcela de aplicaciones industriales que no tienen unas restricciones tan estrictas como las descritas, pero que igualmente pueden aprovechar la metodología de desarrollo aplicable a los **STR**. Esta categoría engloba a la mayor parte de los dispositivos de actuación o interfaz humana, control y supervisión de electrodomésticos, vigilancia desatendida, e incluso algunos tipos de aplicaciones recreativas entre otras. Por supuesto estos no tienen unas restricciones temporales tan críticas como los **STR** anteriormente comentados, satisfaciendo su especificación simplemente con una respuesta interactiva, o lo suficientemente rápida de media. Por tanto, aunque con menos intensidad, también es objetivo de la Informática Industrial, atender a otros aspectos al margen del puro control, como pueda ser la interactuación entre el sistema y el operario. A menudo las máquinas industriales muestran diferentes aspectos del estado interno con el único propósito de informar al exterior, de forma que dejan a juicio del operario la toma de decisiones más trascendentales.

La Informática Industrial pretende ayudar a resolver estos problemas mediante soluciones informáticas, ya que se muestra muy adecuada ante problemas intrincados, y ante problemas no lineales. En otras palabras, el controlador estará formado por un sistema digital programable que ha de ofrecer una respuesta correcta dentro del margen de actuación predefinido. Uno de los objetivos prácticos a perseguir es que no parezca que existe realmente un sistema informático, a excepción de los casos ya aludidos como los *Sistemas de Interfaz Humana*. El controlador debe mostrar un comportamiento similar al que tendría un sistema puramente electrónico, mecánico o físico, en el que

a cada acción o consigna corresponde una reacción, o un cambio de estado, en los momentos oportunos.

Existen muchas formas para definir un Sistema de Tiempo Real. Quizá la más aceptada sea la siguiente:

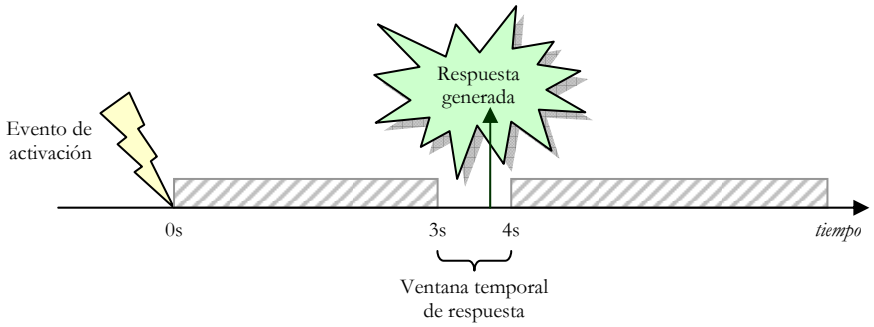
"Un sistema de tiempo real es aquel en el que su funcionamiento es correcto si no depende únicamente de un resultado computacionalmente correcto, sino que depende además del tiempo en el que se haya producido el resultado. Si no se cumplen las restricciones temporales del sistema, se dice que ha ocurrido un fallo en el sistema."

Por tanto, un STR además de no tener fallos de diseño, desarrollo, implementación, etc., debe realizar algún tipo de labor dentro de una ventana temporal. Esta respuesta debe ser predecible e independiente de otras actividades que el sistema pueda estar realizando. Una implicación es que los tiempos de respuesta del controlador han de ser conocidos en los peores casos ya que son los más restrictivos. Los STR a menudo emplean dispositivos de hardware específicos con controladores de dispositivo especiales, que liberan de responsabilidades a los posibles sistemas informáticos que pudiera contener.

Además de respetar estas restricciones temporales para las tareas de tiempo real, a un sistema informático también le exigiremos que use el hardware lo más eficientemente posible, y que optimice la utilización de la CPU para dar el mejor soporte posible al resto de tareas.

Los controladores usados en aplicaciones industriales están muy orientados hacia la automatización de procesos. Las máquinas no sólo han de tomar las decisiones correctas, sino que además tienen que interactuar sincronizadamente entre sí. Ya que estas actúan con un comportamiento físicamente predecible, es posible por ejemplo determinar las trayectorias que llevarán sus piezas móviles. Querríamos que se ordenaran las actuaciones en los instantes precisos teniendo presente la cinemática de los bólidos; en caso de no hacerlo se habría fracasado.

A continuación se muestra una figura que muestra la ventana temporal en donde pretendemos que el sistema actúe.



El rayo representa un evento proveniente posiblemente de algún sensor o de un temporizador. En el ejemplo se exige que la respuesta sea ofrecida dentro del intervalo de tiempo comprendido entre los 3 y 4 segundos transcurridos desde entonces. Si la respuesta fuera ofrecida pasados los 4 segundos, el sistema habría fallado. Igualmente, si se ofrece con anterioridad a los 3 segundos, también se consideraría como un fallo. Obviamente es mucho más fácil asegurar que se respete el límite de la cota mínima que el de la máxima, debido a que es muy sencillo introducir retrasos intencionados. Dependiendo del sistema, es posible que exista más de una ventana temporal durante la que esté permitido ofrecer la respuesta ante un mismo evento.

Como ejemplo, consideremos una máquina que pone chapas a botellines de cerveza. El botellín se coloca en el lugar adecuado tan solo durante un intervalo de tiempo limitado, durante el

cual el tapón debe quedar puesto. De nada sirve que la máquina intente poner el tapón después de que el botellín haya pasado, e igualmente sería absurdo que lo intentara con anterioridad. La máquina debe actuar en perfecta sincronía con el conjunto. En este caso tenemos un ejemplo de los llamados *Sistemas de Tiempo Real no estrictos* o blandos (*soft*). En estos se puede asumir una cierta tasa de fallos, que se traducirían en una mínima pérdida económica, como la producida si se fallara por ejemplo con una botella de cada cinco millones. En tales casos, también se suele disponer de algún mecanismo para detectar el fallo a posteriori, y eliminar el elemento defectuoso.

Existe no obstante otro tipo de sistemas que no admiten fallos en el cumplimiento de sus especificaciones. Estos son los *Sistemas de Tiempo Real estrictos* o duros (*hard*). Entre ellos y atendiendo a su finalidad, podemos encontrar por una parte, a los que por un fallo pueden poner en peligro la vida o la integridad humana o desencadenar grandes pérdidas económicas, y por otra parte, a los destinados a herir, matar y destruir con eficacia y selectividad. Suelen ser mucho más difíciles de desarrollar y en general dependen en mayor medida de sistemas no informáticos. Se debe procurar que la complejidad del sistema sea la menor posible, y resolver los problemas de las formas más seguras y efectivas.

Un ejemplo de STR estricto es el sistema de control de un avión de pasajeros, compuesto a su vez de multitud de subsistemas igualmente críticos. En un momento determinado, un fallo que provocara un retraso en la actuación del sistema de presurización, podría implicar consecuencias graves para la empresa en términos económicos, y posiblemente para la salud de los pasajeros. La mayoría de los sistemas de seguridad presentes en máquinas móviles son también por lo general STR estrictos: en ocasiones es necesario detener una máquina en el instante en que se detecte que puede causar daños de consideración, o llevarla al menos a un estado seguro.

Entre los sistemas que no tienen restricciones de tiempo real podemos encontrar aquellos de los que se espera una respuesta interactiva, o simplemente lo más rápida posible. Además de los encargados de informar, podemos clasificar de este tipo a todas las aplicaciones cuyo objetivo principal se basa en el procesamiento de información, pero en las que no se especifica un plazo de respuesta, como por ejemplo los simuladores.

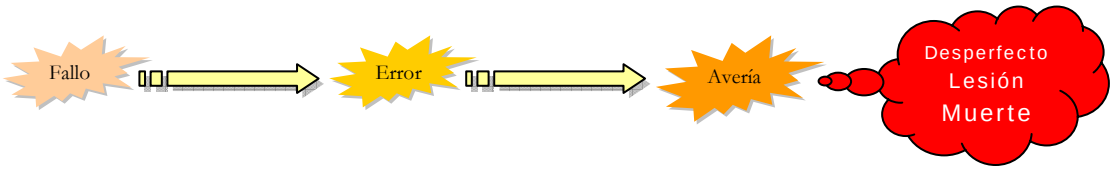
Nota

Al igual que otras acepciones informáticas, el concepto de **Sistema de Tiempo Real** no es totalmente universal, teniendo definiciones o interpretaciones ligeramente diferentes según autores. Por ejemplo se habla se **STR** para información bursátil, actualización de bases de datos, u otros, en los que la latencia del sistema sea mínima. En este contexto se suele decir entonces que *un ordenador ejecutando* ***Windows NT*** *constituye un* ***Sistema de Tiempo Real blando***. En esta asignatura no se va a considerar este tipo de interpretaciones, y no se va a considerar a ningún computador ejecutando un **Sistema Operativo de propósito general** como un **Sistema de Tiempo Real.**

Fiabilidad

Un factor común que ha de tener cualquier STR es una especificación clara, completa, y concisa que no dé cabida a ambigüedades. A partir de dicha especificación, para poder implementar correctamente el controlador se ha de tener un conocimiento suficiente del comportamiento de la planta, de los posibles agentes externos, y del comportamiento esperado del controlador en diferentes circunstancias. La *fiabilidad* de un sistema intenta medir la conformidad real del mismo respecto a su especificación; en el mejor de los casos estaría avalada por variables estadísticas que la definiera y clasificara según una serie de pruebas y comprobaciones objetivas.

En ocasiones, el mal funcionamiento de un STR se debe a *errores* algorítmicos, mecánicos, eléctricos, o a desarrollos informáticos deficientes. Las causas que los provocan se llaman *fallos*. Cualquier STR bien diseñado debería ser *tolerante a fallos*; es decir, si se hubiera introducido un fallo durante el desarrollo que ocasionara un error, el sistema ha de ser capaz de reponerse y seguir funcionando lo mejor posible. A menudo los *errores* internos de implementación se manifiestan en forma de *averías* que hacen que la salida difiera respecto al comportamiento esperado según la especificación. Lo ideal es que el sistema de programación ya sea el lenguaje, el compilador, las bibliotecas, etc., permita la detección y captura de errores de tiempo de ejecución antes de que causen averías y se llegue a una situación desastrosa e irreparable.



Otra forma de mejorar la fiabilidad del sistema consiste en la *prevención de fallos*, que pretende evitar que se introduzcan fallos en el sistema antes que este pase a funcionar. Para ello se suele simular o emular el funcionamiento del sistema completo o incluso, se hacen pruebas en entornos pseudo-reales. En ambos casos se exige un estudio previo de los potenciales fallos que pueden ocurrir. Otras formas de prevenir fallos consisten en: el desarrollo metodológico realizado por personal experto, escritura elegante de código, correcta documentación, uso de herramientas adecuadas para la generación de programas, uso de técnicas de ingeniería de software, etc. No se tienen en cuenta en este contexto los fallos de codificación que se eliminen en tiempo de compilación, ya que no llegan a repercutir a la aplicación final. Por supuesto también colabora en la prevención de fallos el uso de un hardware fiable de alta calidad, correctamente ensamblado y protegido para preservarlo de la hostilidad del ambiente presente allí donde funcione. De especial interés es el uso de herramientas que permitan eliminar fallos en la fase de desarrollo mediante técnicas de depuración. Cabe incidir en una observación importante: cualquier procedimiento destinado a eliminar fallos de un sistema, en realidad tan sólo es capaz de demostrar la existencia de los mismos, pero no su ausencia.

Pese a estas indicaciones, hemos de comprender que ni el mejor STR funcionará para siempre. Por ello es necesario justificar y documentar adecuadamente los márgenes de funcionamiento en los que el sistema funcionará correctamente, así como la duración del mismo (su vida útil), y las necesidades de mantenimiento si las hubiera.

Un ejemplo de STR especialmente fiable es el del fusible eléctrico. Éste es un clásico elemento de protección contra sobreintensidades en el hogar y en la industria. Se trata de un conductor eléctrico realizado normalmente con un fino hilo de plomo por el que pasa una corriente. Si ésta sobrepasa un valor determinado, el conductor se funde debido al sobrecalentamiento, interrumpiendo así la consiguiente circulación de energía eléctrica. Ya que su funcionamiento se basa principalmente en una ley fundamental de la física, su complejidad es ínfima, siendo entonces su seguridad muy elevada. No obstante, es interesante observar que por motivos prácticos en muchas aplicaciones de protección, el fusible ha sido desplazado por el interruptor automático magneto-térmico, más caro, mucho más complejo, y por tanto menos fiable. Esto es así debido a que los usuarios terminan haciendo un mal uso de los sistemas de protección basados en clásico fusible, los que por desconocimiento o dejadez acaban siendo sustituidos por puentes eléctricos mediante hilos conductores que podría ser que no llegaran a fundirse ni siquiera en caso de cortocircuito. Como vemos en muchos casos de la vida real la protección con interruptores automáticos magneto-térmicos es más fiable que el uso de fusibles, pese a que no lo son desde el punto de vista objetivo de su especificación.

Sistemas de Tiempo Real y Sistemas empotrados (*embedded*)

De lo que se ha visto hasta este punto se deduce que la gran mayoría de STR se diseñan para aplicaciones específicas. El entorno en el que deben funcionar por lo general se debe integrar con el resto del sistema o cuando menos, pasar desapercibido. Para que esto sea posible, los STR deben ser lo suficientemente pequeños y discretos. Además en el caso de que no puedan disponer de alimentación a la red eléctrica, su consumo ha de ser muy bajo para poder funcionar con algún tipo de acumulador de energía. A un sistema de estas características se le llama *empotrado* con el significado de *contenido* o que *forma parte de* un sistema de mayores dimensiones. No todos los STR son empotrados, ni un sistema empotrado ha de ser necesariamente de TR, no obstante estos dos términos forman un binomio casi inseparable ya que a menudo si un sistema es una cosa, también es la otra simultáneamente.

Los STR que atañen a la Informática Industrial conllevan cierto procesamiento de información, que puede llegar a ser de una magnitud considerable. En ocasiones se dice coloquialmente que añaden *inteligencia* a un sistema.

Por lo general son sistemas *reactivos*, que realizan funciones en respuesta a eventos, pe. el mando a distancia de una grúa hace que se eleve un garfio al accionar cierto botón. El sistema reacciona ante los agentes externos al mismo siguiendo unas reglas de comportamiento relativamente simples.

En los STR en los que no exista procesamiento masivo de información, se puede emplear microcontroladores o microprocesadores de bajas prestaciones (pe. el 68HC11, 8051,…). Por otra parte, hay otro tipo de sistemas, normalmente de control o procesamiento avanzado, en los que se deben realizar enormes cantidades de operaciones de cálculo numérico. Ejemplo de tales sistemas son la regulación de velocidad de motores eléctricos, y en general la regulación automática de sistemas. En estos casos es normal el uso de algoritmos matemáticos avanzados que implican el cálculo de: Transformadas de Fourier, Convoluciones, Correlaciones, Filtros (FIR, IIR, PID,…), y el Control Digital en Variables de Estado, entre otros. Es por ello que se haga necesario acudir al uso de unos procesadores especializados en ese tipo de cálculo, llamados *Procesadores Digitales de Señal* (DSP), o en su defecto de otro tipo de micros cuyas características computacionales sean igualmente avanzadas.

Ejercicios

1. Enumera 4 ejemplos de STR blandos y otros tantos duros. Especifica en cada uno de ellos por qué lo es.

2. ¿Es un ser humano un STR?

3. Se dice que un sistema es *seguro* si no se pueden producir situaciones que puedan dañar, matar, ni estropear o destruir bienes materiales o ambientales, que supongan una pérdida significativa. A tenor a esto, ¿Son todos los STR seguros?